VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Komunikační systém pro varování osob před nebezpečím s využitím IP telefonie

IP telephony Based Danger Alert Communication System

2012 Filip Kolenovský

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Filip Kolenovský**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Komunikační systém pro varování osob před nebezpečím s využitím IP telefonie
IP telephony Based Danger Alert Communication System

Zásady pro vypracování:

Systémy určené pro varování širokého okruhu obyvatelstva před nebezpečím či sloužící jako informační zdroje jsou v současnosti poměrně rozšířené a používají se v různých oblastech lidské činnosti. Většina z těchto systémů však pracuje s rozesíláním textových zpráv či distribucí zprávy pomocí centrálního zařízení (sirény, rozhlas apod.). Cílem práce je vytvořit systém s využitím technologií paketově přepínaných sítí se službou VoIP a protokolem SIP pro generování a distribuované rozesílání hlasových zpráv přes ústřednu přímo na koncové zařízení (mobilní telefon, pevná linka, atd.).

1. Studijní část: Linux, MySQL, Apache, PHP, SIP, Sipp.
2. Požadavky na komunikační systém a omezení vyplývající z reálného použití.
3. Návrh algoritmu pro distribuci hlasových zpráv.
4. Realizace navržených algoritmů ve formě webové aplikace.
5. Testování systému a porovnání reálných výsledků z předpokládanými.
6. Vytvoření uživatelské a programátorské dokumentace komunikačního systému.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího diplomové práce.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Filip Řezáč**

Datum zadání: 18.11.2011

Datum odevzdání: 04.05.2012

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky * OSTRAVA *

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.*

V Ostravě 15. dubna 2012 .....

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 15. dubna 2012 .....

Rád bych na tomto místě poděkoval Ing. Filipu Řezáčovi za odbornou pomoc a konzultaci při vytváření této diplomové práce.

Abstrakt

Cílem této práce je vytvořit webovou aplikaci umožňující rozesílat přednahrané zprávy pomocí VoIP telefonie. Nejdříve dojde k seznámení s technologiemi použitými při vytváření aplikace, a následně k rozboru a popisu implementace algoritmu použitého pro rozesílání zpráv. Poslední fáze obsahuje porovnání výsledků dosažených v reálné aplikaci s předpokládanými hodnotami. Díky podrobnému popisu konfigurace jednotlivých použitých služeb lze jednoduše nasadit a replikovat danou aplikaci pro účely reálného použití.

Klíčová slova: SIP, SIPp, Voip, Asterisk, Php

Abstract

The aim of this thesis is to develop web application, that allows to send pre-recorded messages by using VoIP telephony. Firstly, the technologies used in application are introduced. Then the analysis and the description of the implementation of the algorithm used for sending messages are presented. In the last stage there is a comparsion of the results achieved in a real application, with the expected values. Due to the detailed description of the configuration of the particular services it is very simple to replicate and deploy the application for real use.

Keywords: SIP, SIPp, Voip, Asterisk, Php

Seznam použitých zkratek a symbolů

HDS	–	Hlasový distribuční systém
HTML	–	Hyper Text Markup Language
XML	–	Extensible Markup Language
SIP	–	Session Initiation Protocol
PHP	–	Hypertext Preprocessor
SQL	–	Structured Query Language
CSV	–	Comma-separated values
IVR	–	Interactive voice response
PCAP	–	Packet capture
WAV	–	Waveform audio file format
BTS	–	Base transceiver station
SHA	–	Secured hash
PDO	–	PHP Data Objects
AJAX	–	Asynchrounous JavaScript and XML
VoIP	–	Voice over IP
PBX	–	Private branch exchange
GPL	–	General public licence
B2BUA	–	Back to back user agent
SMS	–	Short message service
OOP	–	Object-oriented programming
PID	–	Process identifier

Obsah

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Téměř každý den se setkáváme s množstvím nehod, katastrof a jiných nebezpečných situací ohrožujících lidský život a majetek. Ať už se jedná o situaci vzniklou přírodními silami či lidskou činností, je vždy důležité, aby lidé nacházející se v místě této situace byli včas informováni před možným nebezpečím, a měli tak možnost učinit opatření k minimalizaci škod a záchraně lidských životů. Pro účely šíření informací je dnes vyvinuto poměrně rozsáhlé spektrum prostředků, jak danou informaci efektivně předat na místa, kde je zapotřebí. Ne vždy se však musí daná situace odehrávat v dosahu těchto informačních prostředků. Zejména na odlehlých místech, na kterých se nenachází žádný rozhlas, sirény ani jiné možnosti hromadného šíření informací, může být včasné dodání potřebné informace problematické.

Nejrozšířenějším způsobem komunikace v dnešní společnosti jsou mobilní telefony. Ve většině civilizovaných zemí mají mobilní operátoři pokrytou velkou část území a mobilní telefon vlastní většina populace, což z něj činí ideální nástroj pro rozesílání upozornění na hrozící nebezpečí. Většina dnešních systémů ovšem pracuje na základě zasílání SMS zpráv, které ovšem mohou být lehce přehlédnuty nebo přečteny se zpožděním. Řešením je zasílání nahraných zpráv přímo na čísla nacházející se v krizové oblasti, a v případě nevyzvednutí zprávy zaslat v určitém intervalu zprávu znovu. Pro rozesílání těchto zpráv vytvoříme aplikaci využívající VoIP telefonie, která umožňuje distribuované rozesílání hlasových zpráv pomocí softwarové ústředny jak na mobilní telefony, tak na pevné linky. Daná aplikace bude založena na open-source technologiích a bude ji tedy možné okamžitě využívat bez nutnosti zakoupení dalších licencí. Aplikace je určena pro pracovníky krizových center, kteří díky ní budou moci okamžitě reagovat na hrozící nebezpečí a okamžitě mohou informovat lidi nacházející se v zasažené oblasti.

Mým úkolem je navrhnout algoritmus distribuce hlasových zpráv, naimplementovat tento algoritmus ve formě webové aplikace s vhodným uživatelským rozhraním, nakonfigurovat potřebné softwarové součásti na testovacích sestavách a provést srovnání reálných výsledků aplikace s předpokládanými. Od vypracování této práce očekávám prohloubení teoretických i praktických znalostí z oblasti VoIP telefonie a vytvoření produktu, který najde uplatnění v reálném provozu.

V jednotlivých kapitolách práce jsou popsány důležité prvky použité během návrhu a tvorby systému. V první části jsou popsány technologie, s jejichž pomocí byl vytvořen. Následně je provedena analýza požadavků a omezení, které jsou limitující pro správnou

činnost aplikace. Důležitý popis algoritmu, jenž provádí samotné generování a zpracování hovorů je umístěn v následující kapitole. Poslední dvě části jsou věnovány implementaci zdrojových kódů aplikace a testování hotového systému, včetně ukázek důležitých konstrukcí. Zdrojové kódy aplikace spolu s programátorskou a uživatelskou dokumentací jsou umístěny v přílohách.

2 Použité technologie

Celý systém je postaven na open-source technologiích, což značně snižuje jeho koncovou cenu. Aplikace je navržena na webové architektuře server-client, využívá kombinaci webového serveru Apache, databázového systému MySQL a jako skriptovací jazyk je použito PHP. Operačním systémem byl zvolen Linux, konkrétně distribuce Debian, pro svou spolehlivost a stabilitu. Další částí potřebnou pro provoz tohoto systému je telefonní PBX ústředna umožňující registraci SIP účtů a zvládající obsloužit velké množství hovorů v jeden okamžik. Pro tento účel byl zvolen software Asterisk, který splňuje všechny podmínky nutné pro distribuované rozesílání hovorů a díky své rozšířenosti nebude problém nasadit jej na téměř jakémkoliv hardware. Dynamické generování hovorů bude obstarávat nástroj SIPp, jenž bude pomocí přednastavených .xml schémat definovat formát SIP zpráv a .csv soubor obsahující cílové uživatele a jejich telefonní čísla, ze kterého se budou dynamicky načítat tyto hodnoty do .xml schémat. Z důvodu rychlejšího a kvalitnějšího vývoje aplikace bude pro její zhotovení využit PHP framework Nette. Toto umožní vytvořit univerzální kód srozumitelný pro široký okruh programátorů a v rámci implementace bude možné využít předpřipravených komponent. Oproti tvorbě aplikace bez frameworku použití osvědčených komponent dokáže značně ulehčit práci a není nutné připravovat základní struktury nutné pro funkčnost aplikace ručně. Nette framework byl vybrán z důvodu bezpečnosti, rychlosti a jednoduchosti, s níž v něm lze webové aplikace vyvíjet.

2.1 Platforma

2.1.1 Linux

Operační systém založen na Unixovém jádře Linusem Torvaldsem v roce 1991. Řídí se standardy POSIX a single Unix specifications, které definují Unixové systémy a kladou důraz operačního systému na jednoduchost. Oproti Unixovým systémům je jádro doplněno o další softwarové vybavení (aplikace, utility, grafické uživatelské rozhraní) nutné pro vytvoření plnohodnotného operačního systému, který je možné využívat nejen na serverech a pracovních stanicích, ale na všech běžných počítačích (desktopové, přenosné, tablety a jiné). Podle softwaru dodávaného s jádrem je Linux rozdělen do různých distribucí (Ubuntu, Debian, Fedora a další), kdy každá svým vybavením klade důraz na různé vlastnosti systému a také jejich vývoj je veden a financován různými způsoby. Hlavní znaky Linuxových systémů jsou stabilita, bezpečnost, otevřenost a variabilita,

mezi důležité vlastnosti poté patří multitasking (víceúlohovost) a víceuživatelský přístup. Většina distribucí spadá pod licenci GNU GPL, která umožňuje jejich volné používání, distribuci a úpravy, některé distribuce (např. Red Hat Enterprise) jsou určeny pro komerční sféru a pro jejich používání je licence nutné zakoupit. (zdroj [8], [9])

2.1.2 Apache

Apache je multiplatformní softwarový webový server, který je možno provozovat na všech rozšířených operačních systémech (Solaris, Linux, Mac OS X, Windows). Je vyvíjen od roku 1993 a momentálně se jedná o nejpoužívanější server v prostředí internetu (63% všech webových stránek). Používá modulární architekturu, kdy přidávání dalších funkcí k základní funkcionalitě jádra je prováděno pomocí instalace modulů (např. mod_auth, mod_rewrite a další). Ačkoliv se nejedná o nejvýkonnější dostupné řešení, je možné využít různých implementací serveru (tzv. MultiProcessing moduly) a přizpůsobit jej tak danému hardware, na kterém je provozován. Nejnovější verze Apache 2.0 využívá vlastní licence, která je však kompatibilní s GPL (general public licence) a server je tedy dostupný k volnému použití. (zdroj [8], [10])

2.1.3 Asterisk

Softwarová telefonní ústředna (PBX) vyvíjena pro Unixové platformy. Jedná se o flexibilní a momentálně nejrozšířenější řešení v oblasti integrovaných telekomunikačních systémů. Tvoří rozhraní mezi telefonním hardware a softwarovými telefonními aplikacemi a umožňuje tak propojení libovolných telekomunikačních zařízení. Asterisk obsahuje spektrum možností pro komunikaci, jako například hlasové zprávy, konferenční hovory, volbu možností pomocí kláves telefonu (IVR), automatickou distribuci hovorů a mnoho dalších. (zdroj [5])

2.1.4 SIPp

SIPp je nástroj pro testování a simulaci provozu SIP protokolu. Umožňuje simulovat reálný provoz generováním a odesíláním požadavků na testovanou telefonní ústřednu, pomocí předpřipravených schémat nebo pomocí .xml souborů nadefinovaných uživatelem. Během simulace toku hovorů na ústřednu umožňuje zobrazit statistiky jako: počet provedených hovorů, doba provádění požadavků, statistiky potvrzujících a chybových zpráv. Důležitou vlastností je možnost zasílání RTP medií ve formě .pcap nahrávky a také

emulace uživatelů pomocí externího .csv souboru, ve kterém mohou být uloženy jejich jména spolu se SIP identifikátory a SIPp postupně zasílá zprávy všem těmto uživatelům. Tento nástroj může být použit pro testování velkého množství zařízení, například SIP proxy, B2BUA (back to back user agent), SIP media serverů, SIP/x brán, PBX ústředen. (zdroj [7])

2.2 Jazyky a protokoly

2.2.1 MySQL

MySQL je relační databázový systém typu DBMS (database managment system). Byl založen švédskou společností MySQL AB, později odkoupen společností Oracle, která jeho vývoj financuje i nyní. Dotazy na databázi jsou prováděny pomocí jazyka SQL (structured query language), doplněným o některé rozšíření. Hlavním cílem vývojářů tohoto systému je vysoký výkon, což má za následek některé zjednodušení, jako jednoduché způsoby zálohování nebo podpora triggerů, uložených procedur a pohledů, které byly přidány až v posledních několika letech (verze 5). Jednotlivé dotazy jsou systémem pro jejich nejrychlejší provádění optimalizovány (například pořadí načítání tabulek). Pro ukládání dat jsou využívány různé úložné enginy (storage engines), například MyISAM nebo InnoDB, kdy každý engine využívá jiný způsob zápisu dat do souborového systému a podporuje jiné funkce (např. cizí klíče, transakce atd.) možné nad těmito daty provádět. Velké rozšíření systému MySQL je zapříčiněno vysokým výkonem, snadnou nasaditelností a licencí GPL, která umožňuje jeho volné šíření. (zdroj [23])

2.2.2 PHP

PHP (Hypertext preprocessor) je skriptovací programovací jazyk využívaný pro tvorbu dynamických internetových stránek a webových aplikací. Jedná se o interpretovaný jazyk, překládání zdrojového kódu tedy probíhá až za běhu programu. Veškerý překlad probíhá na straně serveru a ke klientovi je zasílán pouze výsledek ve formě statické HTML stránky, zdrojový kód aplikace na straně klienta není vůbec zobrazen. Od verze 4 je podporováno objektově orientované programování, avšak plnohodnotné OOP bylo přidáno až od verze 5. Není zapotřebí deklarovat datový typ proměnných, syntaxe jazyka je založena na jazycích Perl, C, Java a Pascal. PHP je nezávislé na platformě, pro jeho běh je zapotřebí server (interpreter), který provádí překlad a běh skriptu (například Apache společně s

PHP modulem). Na straně klienta poté dostačuje jakýkoliv webový prohlížeč. (zdroj [16], [11])

2.2.3 SIP

SIP (Session Initiation Protocol) je internetový protokol pro inicializaci relací. Jeho hlavní využitím je přenos signalizace v internetové telefonii, konkrétně se jedná o lokalizaci, zjištění stavu a možností účastníka, navázání a řízení spojení. V základním nastavení využívá UDP port 5060, avšak tento port může používat i nad protokolem TCP. Vychází z protokolu HTTP a využívá také některé funkce protokolu SMTP, přičemž jeho hlavní výhodou oproti staršímu protokolu H.323 je jednoduchost. Jedná se o textově orientovaný protokol, příkazy jsou zapisovány velkými písmeny (REGISTER, INVITE, ACK), odpovědi na požadované příkazy jsou zasílány ve formě čísla odpovědi, společně s textem (200 - OK, 100 - Trying). (zdroj [2], [4])

3 Požadavky a omezení reálného použití

3.1 Požadavky zadavatele

Prioritou při návrhu vyvíjeného systému byla co nejjednodušší možnost jeho nasazení a používání. Při porovnání faktorů, které schopnosti nasaditelnosti ovlivňují nejvíce, bylo zjištěno, že nejdůležitějším faktorem je v tomto případě výsledná cena za potřebný hardware a také softwarové licence. Z tohoto důvodu bylo v návrhu zadavatele aplikace požadováno použití webové architektury Klient-Server, která umožňuje k dané aplikaci přistupovat z jakékoliv platformy, takže používání systému není omezeno operačním systémem ani výkonem dané stanice. Jediný požadavek na straně klienta je webový prohlížeč a funkční síťové spojení k serveru. Při výběru použitých technologií pro implementaci systému je taktéž kladen vysoký důraz na cenovou dostupnost, a proto jsou veškeré platformy (operační systém, databáze i programovací jazyk) zvoleny s ohledem na licenci, pod kterou jsou dostupné, i za cenu složitější implementace oproti komerčním řešením. Veškerý použitý software je volně dostupný pod open-source licencí, a není tedy nutné za jejich použití platit žádné poplatky. Hlavním výdajem pro nasazení systému je tedy pořizovací cena aplikačního serveru a cena SIP proxy serveru (ceny klientských počítačů jsou v tomto případě zanedbatelné). Při použití komerčních technologií (například při použití asp.Net spolu s databází MS SQL) by výsledná cena byla podstatně vyšší a nasazení do provozu obtížnější.

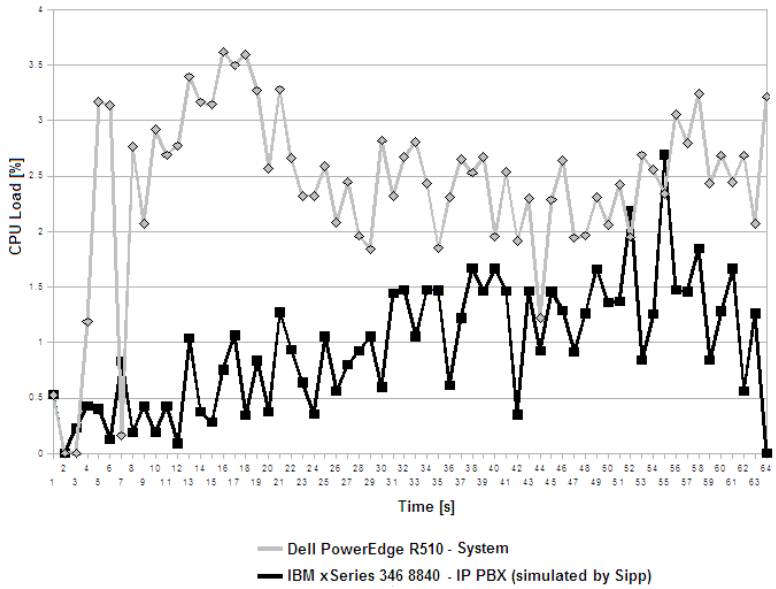
3.2 Omezující faktory

Hlavní otázkou, která určuje efektivitu systému je: kolik SIP zpráv a hovorů je systém schopen provést v jednom okamžiku? Pro určení této hodnoty je nutné prověřit hned několik faktorů:

1. Výpočetní nároky na server se systémem
2. Výpočetní nároky na server s ústřednou
3. Šířka přenosového pásma na lince mezi systémem a ústřednou
4. Maximální zatížení aplikace SIPp

Tyto omezení byly prakticky otestovány pracovníky katedry telekomunikací ([1, článek konference]) a z výsledků testování byly zjištěny následující skutečnosti. První 2

faktory jsou značně závislé na použitém hardware a je možné v případě potřeby zvýšit propustnost nasazením výkonnějších komponent. Během testu byla simulována zátěž pomocí generování 500 SIP požadavků spolu s 60s přednahranou zprávou v jednom okamžiku. Test probíhal na serverových systémech Dell PowerEdge R510 jako systémovým serverem a IBM xSeries 346 8840 s aplikací SIPp simulující chování ústředny. Z naměřeného grafu (1) vidíme, že procentuální zatížení procesorů obou stanic nepřesáhlo 4% a můžeme tedy usuzovat, že výpočetní nároky na oba servery nebudou limitovat maximální počet generovaných hovorů. Z důvodu bezchybného chodu celého systému je však vhodné použití dvou oddělených fyzických strojů, pro ústřednu a webovou aplikaci zvlášť.



Obrázek 1: Zátěž CPU (zdroj [1])

Šířka přenosového pásma je závislá na velikosti datového toku zasílaných předehraných zpráv. Tuto hodnotu můžeme ovlivnit volbou použitého kodeku, pomocí kterého bude zpráva kódována. Datové toky jednotlivých kodeků můžeme vypočítat podle vzorce (1), kde h je velikost všech hlaviček, C_r je přenosová rychlost kodeku a Δt_s je rozestup mezi packety se vzorky. Tato hodnota lze zjistit ze vzorce (2), kde P_s je užitečná zátěž a C_r přenosová rychlost.

Počet hovorů	Kodeky a jejich šířka pásma [Mbit/s]			
	G.711	G.729	G.723-ACELP	GSM
100	9,04	3,44	2,4	4
500	45,2	17,2	12	20
1000	90,4	34,4	24	40

Tabulka 1: Šířka pásma jednotlivých kodeků

$$BW = \frac{(8 * h + C_r * \Delta t_s)}{\Delta t_s}[kbit/s] \tag{1}$$

$$\Delta t_s = \frac{P_s}{C_r}[ms] \tag{2}$$

Výpočet jsme provedli pro 4 nejpoužívanější kodeky a hodnoty 100 až 1000 hovorů. Z tabulky vypočtených hodnot můžeme vypozorovat, že rozdíly datové náročnosti jednotlivých kodeků jsou poměrně velké. Například při použití datově nejobjemnějšího kodeku G.711 bychom při 1000 současných hovorech dosáhli datového toku 90,4 Mbit/s, což je již prakticky maximální limit běžné 100Mbit/s linky. Naopak nejúspornější kodek G.723-ACELP by pro přenos stejného počtu hovorů potřeboval pouze 24Mbit/s a při jeho použití by k přiblížení se limitu 100Mbit/s došlo až při zprostředkování 4000 hovorů. Z těchto hodnot tedy můžeme usuzovat, že šířka pásma může limitovat konečný počet generovaných hovorů.

Faktorem číslo 4, který ovlivňuje množství generovaných hovorů v jeden okamžik, je maximální zatížitelnost aplikace SIPp. Během testů této aplikace se zjistilo, že právě SIPp nejvíce limituje celý systém, protože dokáže bezchybně vygenerovat maximálně 700 SIP požadavků. Nad tímto limitem docházelo k zasílání chybně strukturovaných nebo nesprávně dlouhých INVITE požadavků. Jako bezpečnou hodnotu systémem generovaných hovorů je tedy určen počet 500 a větší množství požadavků v jednom momentu není možné uskutečnit.

Pro rozesílání zpráv do oblastí, kde je více než 500 lidí je tedy nutné cílové uživatele rozdělit do skupin (grup) po 500 účastnících. Je ovšem nutné určit, jaký časový interval zvolit mezi obsluhou jednotlivých grup. Pokud určíme maximální délku přednahrané zprávy na 30s, což je standardní limit pro zprávy hlášené v elektronických sirénách, k tomu připočteme dobu vyzvánění 15s, tak dostaneme hodnotu 45s na jeden hovor. Pokud

zanecháme rezervní dobu, například pro obsluhu nepřijatých hovorů a dalších režijních činností, které mohou nastat, pak můžeme jako vhodnou dobu pro obsluhu jedné skupiny určit čas 60s.

V průběhu implementace jádra pro distribuci hovorů byla aplikaci SIPp týmem Liptel přidána optimalizace pro vícevláknový provoz. Jedno procesorové jádro je takto schopno bezchybně generovat a udržovat 200 SIP požadavků v jeden okamžik. Jelikož však již bylo jádro aplikace z velké části dokončeno, zůstala aplikace určena pouze pro jednovláknové využití. K budoucímu využití více vláken však stačí aplikaci doplnit o modul, který bude dané grupy aktivně přidělovat procesorovým jádrům podle jejich dostupného počtu.

4 Návrh algoritmu

4.1 Hlasový distribuční systém

Systém pro hlasovou distribuci zpráv můžeme rozdělit do několika částí a modulů. První část je SIP proxy server, v tomto případě PBX Asterisk, který zajišťuje provedení SIP požadavků a realizaci hovorů k účastníkům. Druhou částí je aplikace SIPp spolu s .xml schématem, podle kterého budou generovány SIP požadavky zasílané na proxy server. Třetí částí je samotná webová aplikace, která je rozdělena do několika modulů. Kostru aplikace tvoří jádro generující parametry předávané aplikaci SIPp spolu s modulem pro distribuci zpráv a tvorbu .csv souborů, ze kterých bude aplikace SIPp dynamicky měnit čísla účastníků v .xml schématu. Další část aplikace tvoří modul pro vstup nahraných zpráv v jiných vstupních formátech, tedy převod psaného textu na řeč a konverze .wav souborů do formátu .pcap. Poslední částí je modul pro ochranu zahlcení BTS, který má za úkol distribuci hovorů na základě zatížení jednotlivých stanic operátora. Mým hlavním cílem je vytvoření funkčního jádra aplikace spolu s modulem pro distribuci zpráv, nakonfigurování asteriskového proxy serveru a vytvoření .xml schématu pro požadované generování požadavků aplikace SIPp. Doplňující moduly pro ochranu zahlcení BTS a převod textu na řeč již nejsou pro funkčnost aplikace zásadní, a s jejich zhotovením se počítá až jakmile bude otestována funkčnost jádra aplikace.

4.2 Požadované vstupy

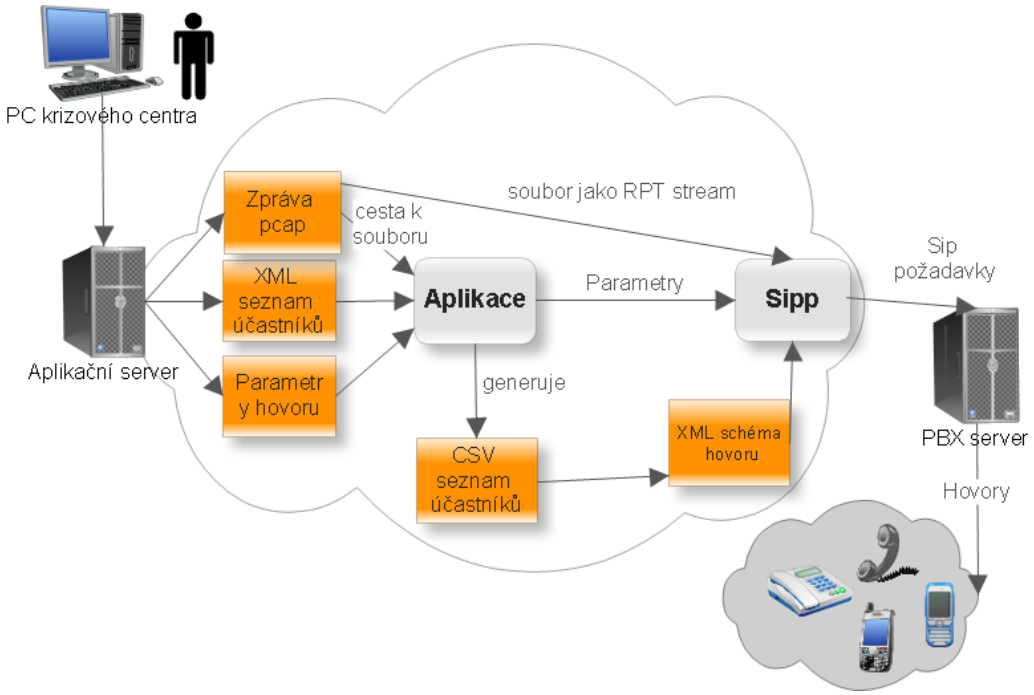
Pro správnou funkci aplikace je zapotřebí spolupracovat s mobilními operátory, kteří pracovníkovi dodají seznam telefonních čísel, jenž se nacházejí v cílové oblasti určené pro distribuci zprávy. Tento seznam je uložen ve formátu .xml a každý účastník má uvedeny 3 atributy: telefonní číslo koncového uživatele, identifikační informace o BTS stanici, ke které je účastník registrován a propustnost stanice BTS. Počet řádků v seznamu se rovná počtu koncových uživatelů a je určen hodnotou C_{max}.

Dalším požadavkem je přednahraná .pcap zpráva, která je rozeslána všem cílovým účastníkům a po vyzvednutí hovoru je přehrána. Pracovník krizového centra může také nastavit dobu vyzvánění hovoru (T_{ring}), po kterou bude účastníkův telefon čekat na přijmutí hovoru, na 5,10 nebo 15 sekund. Pokud do této doby cílový účastník hovor nepřijme, bude považován za nevyzvednutý. V případě, že by nezastižený účastník provedl hovor zpět na zmeškané číslo, bude mu přehrána interaktivní hlasová služba (IVR), která

nabídne volajícímu vybrat si z možných akcí pomocí klávesnice telefonu, přičemž jedna z možností bude přehrání varovné zprávy.

V případě reálné hrozby má pracovník nadefinovány časové modely, ve kterých musí dopravit varovnou informaci k ohroženým osobám. Tento údaj je reprezentován proměnnou T_{max}, která slouží jako limit pro rozeslání všech hovorů. Je možno nastavit také proměnnou callRepeat, která určuje počet opakovaných volání v případě nevyzvednutí hovoru.

Pro správnou distribuci zpráv musí být nastaveny tyto vstupní proměnné:



Obrázek 2: Schéma systému

- C_{max} - počet požadavků splnitelných najednou (limit 500)
- T_{ring} - čas, po který má uživatelům telefon vyzvánět než bude zpráva označena za nedoručenou
- T_{max} - maximální čas, po kterou může aplikace rozesílat hovory (zda je tento čas splnitelný závisí na počtu cílových účastníků)
- callRepeat - limit počtu opakování hovorů v případě nevyzvednutí hovoru

- IP$_{pbx}$ - ip adresa PBX ústředny, která bude zpracovávat SIP požadavky generované aplikací SIPp
- SIP$_{name}$, SIP$_{pass}$ - uživatelský účet, pomocí nějž se bude aplikace SIPp registrovat k PBX

Z výše uvedených vstupních proměnných a souborů poté aplikace vypočítá a obsluhujícímu pracovníkovi zobrazí hodnoty těchto proměnných:

- C$_{req}$ - celkový počet požadavků
- G$_n$ - počet grup, do kterých jsou uživatelé rozděleni
- T$_{snd}$ - předpokládaný čas potřebný k provedení všech hovorů
- T$_{rem}$ - čas zbývající k dovršení T$_{max}$ po dokončení všech požadavků ($T_{max} - T_{snd}$), lze využít k opakovanému volání nedostupným uživatelům.

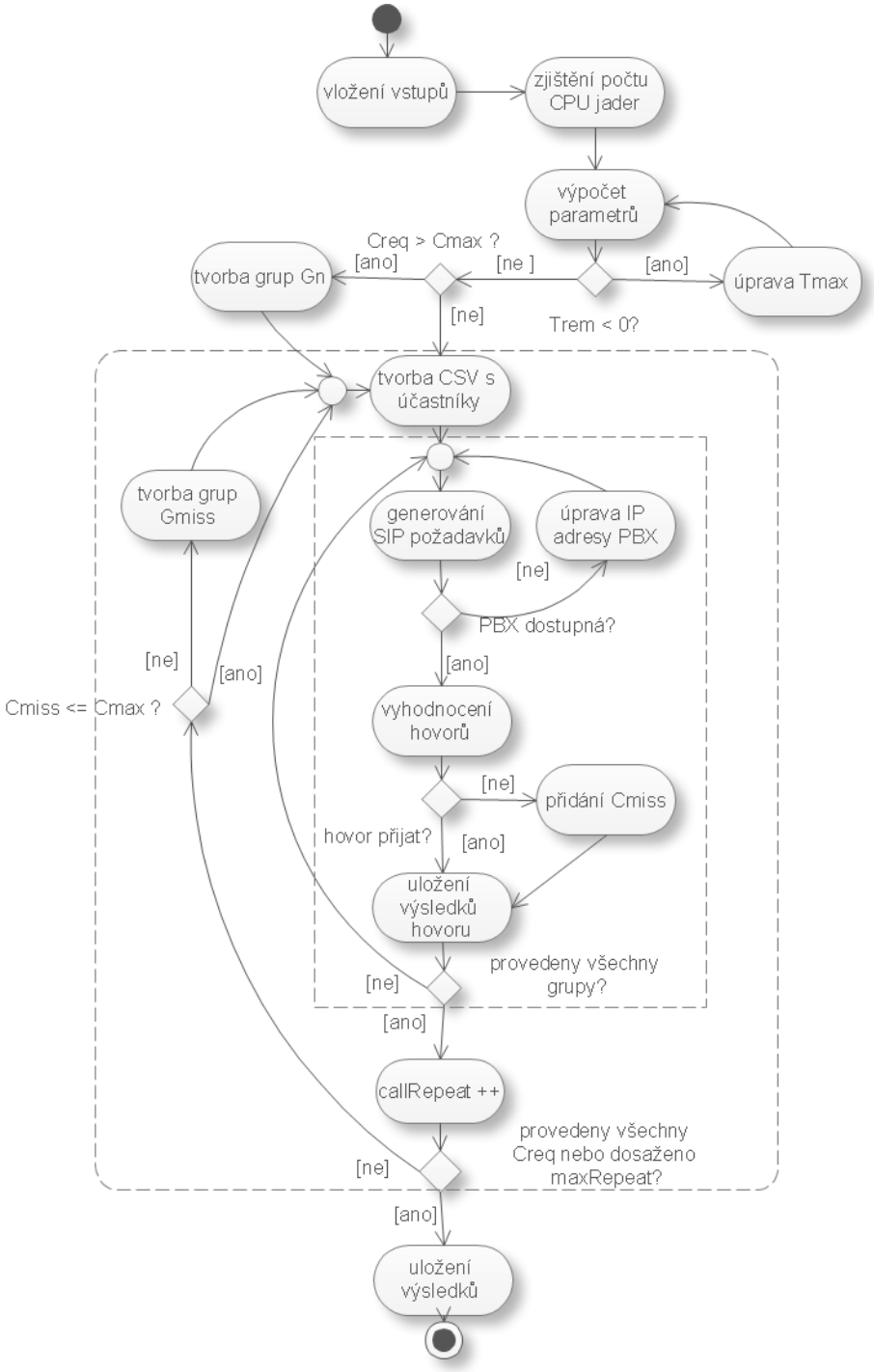
Po provedení všech požadavků C$_{req}$ je proveden výpočet nových požadavků, které v prvním kole nebyly vyzvednuty:

- C$_{miss}$ - požadavky, které nebyly cílovým uživatelem vyzvednuty
- G$_{miss}$ - grupy, do kterých jsou rozděleny požadavky C$_{miss}$

4.3 Algoritmus

Po zadání vstupních proměnných jsou uživatelé rozděleni do grup o velikosti C$_{max}$ a v tomto rozdělení jsou uloženi do databáze, přičemž současně jsou vygenerovány csv soubory obsahující jednotlivé grupy. Z počtu grup (G$_n$) a doby pro odeslání jedné grupy (60 s) je vypočítána předpokládaná doba pro odeslání hovorů (T$_{snd}$). Následně je vypočtena doba ($T_{rem} = T_{max} - T_{snd}$), která zbývá pro rozeslání nedoručených hovorů (C$_{miss}$). Pokud je hodnota T$_{rem}$ záporná, pak není dostatek času pro uskutečnění všech požadavků a pracovníkovi je zahlášena chyba, a je nutné upravit zadávaná data. Pokud záporná není, je pracovníkovi zobrazen přehled vypočtených hodnot a zobrazen časový plán distribuce zpráv.

Pracovník může vstupní hodnoty upravit a nechat přepočítat, uložit pro pozdější použití, anebo rozesílání zpráv spustit. Po spuštění aplikace jsou veškeré parametry uloženy



Obrázek 3: Aktivitní diagram

do databáze a poté předány spolu s .csv seznamem účastníků a předpřipraveným .xml schématem aplikaci SIPp, která začne generovat a zasílat SIP požadavky na server s telefonní ústřednou. Ústředna poté zajistí realizaci hovoru přímo k cílovému účastníkovi a v případě, že hovor příjme, mu přehraje nahranou zprávu. Po provedení dané grupy (G_i) jsou data vrácena z aplikace SIPp analyzována a požadavky rozděleny do skupin s přijatými hovory (C_{ans}) a nepřijatými (C_{miss}) a záznamům v databázi je přidělen stav podle toho, zda byly dostupné nebo ne. Pokud tedy zbývá čas T_{rem} a je povoleno opakované volání nedostupným účastníkům, pak po provedení všech původních grup jsou vytvořeny nové grupy (G_{miss}), které obsahují telefonní čísla jenž hovor v prvním kole nepřijaly. Takto algoritmus pokračuje, dokud není dosažen maximální čas T_{max} nebo dokud není dosažen limit maximálního počtu opakovaných hovorů.

Po skončení běhu algoritmu jsou veškeré údaje uloženy do databáze a pracovníkovi jsou zobrazeny podrobné statistiky z dokončené události. Kdykoliv v průběhu distribuce zpráv má obsluha aplikace možnost běh přerušit. Pokud je přerušení provedeno, má obsluha možnost zadat poznámku s vysvětlením důvodu přerušení. Jednotliví pracovníci mají stejné pravomoci, pouze role vedoucího oddělení umožňuje přidávání nových účtů a případně resetování zapomenutých hesel jiných účtů. Dostupné operace jednotlivých rolí jsou zobrazeny v use case diagramu (obrázek 4). (zdroj [1])

4.4 Jádro aplikace

Hlavním cílem webové aplikace je vytvoření rozhraní mezi obsluhou krizového centra a aplikací SIPp. Proto je hlavní funkcionalita zaměřena na transformaci zadaných požadavků do podoby, která je zapotřebí pro správné spuštění této aplikace a také získávání výsledných dat z jejího běhu. Jádro má tedy za úkol výpočet parametrů předávaných aplikaci SIPp, ovládání této aplikace a zajišťuje vzájemnou komunikaci mezi jednotlivými moduly. Uchovává tedy veškeré proměnné a cesty k souborům dále používané při volání SIPp.

4.5 Modul distribuce zpráv

Vyvíjená aplikace bude dodávat čísla cílových uživatelů, rozdělené do skupin o maximálně 500 záznamech, které budou uloženy ve formátu .csv. Vstupní seznam uživatelů (ve formátu .xml) je tedy nutné rozdělit a transformovat do několika .csv souborů, které budou poté sloužit pro dynamickou změnu parametrů v SIP požadavku. Souběžně s

přidáváním uživatelů do .csv souboru jednotlivých grup, jsou účastníci také ukládáni do databáze, kde jsou také uvedeny veškeré údaje o grupě, ve které se nachází. Tento modul má také za úkol zpracovávat výsledky hovorů a upravovat záznamy uživatelů podle úspěšnosti vyřízení hovorů, případně jejich přiřazení do grup G_{miss}.

4.6 Další moduly

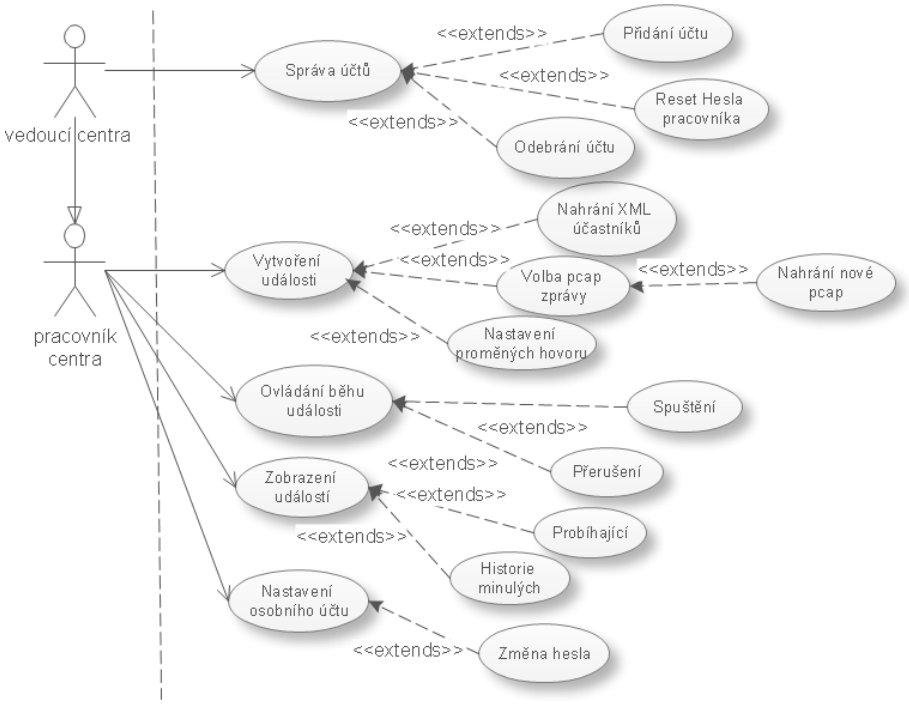
Aplikace SIPp je schopna pracovat s nahrávkami pouze ve formátu .pcap, který není pro nahrávání a editaci zpráv příliš vhodný. Jádro webové aplikace je navrženo pro práci pouze s tímto typem nahrávek. V budoucnu je počítáno s tvorbou modulu pro převod textu na řeč (Text to Speech) a také převod z dostupnějšího formátu .wav na .pcap, který obsluze značně ulehčí tvorbu a manipulaci nahrávek.

Údaje o BTS jsou v seznamu účastníků uvedeny pro potřeby modulu ochrany přehlcení BTS, který bude určovat složení jednotlivých grup tak, aby na BTS v dané oblasti nebylo zasíláno více požadavků než je jejich uvedená propustnost. Tento modul bude spolupracovat s modulem pro distribuci zpráv a jednotlivé grupy budou tvořeny s ohledem na kapacitu využívaných BTS. Pokud by nebylo možné jednotlivé uživatele rozčlenit do grup tak, aby splňovaly požadavky na zatížení stanic operátora, bude automaticky snížena hodnota C_{max}, na hodnotu, která bude určena maximálním množstvím požadavků, které je schopna daná stanice zpracovat.

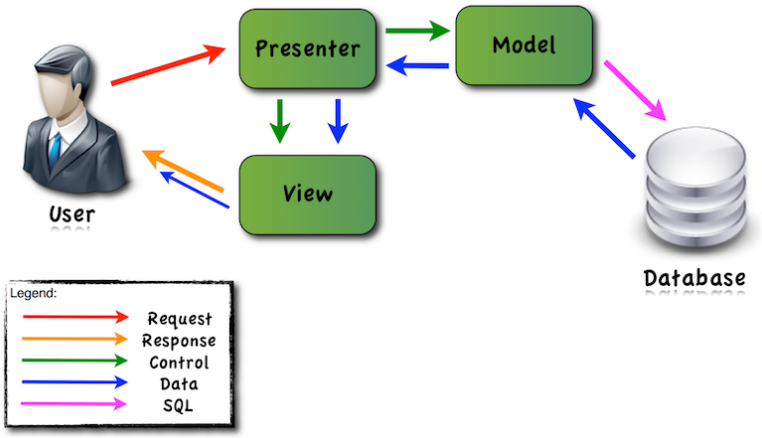
4.7 Architektura Model-View-Presenter

Tento architektonický vzor umožňuje oddělit datovou vrstvu, prezentační vrstvu a logickou vrstvu nezávisle na sobě, takže výsledný kód je přehlednější a případné změny jedné části pouze minimálně ovlivní části ostatní ([24, model view presenter]). Jednotlivé vrstvy jsou odděleny, a je tedy možné, aby na nich pracovaly nezávisle na sobě různé týmy vývojářů. Tato architektura je odvozena ze vzoru Model-View-Controller, avšak zachycení a zpracování některých událostí (například zmáčknutí tlačítka klávesnice, rolování kolečka myši atd.) má za úkol View, zatímco v MVC má veškeré řízení na starost Controller.

1. vytvoření požadavku v prohlížeči
2. presenter získá akci z view
3. presenter zašle požadavek modelu



Obrázek 4: Diagram případů užití



Obrázek 5: Model-View-Presenter (zdroj [24])

4. model aktualizuje svá data na základě požadavku

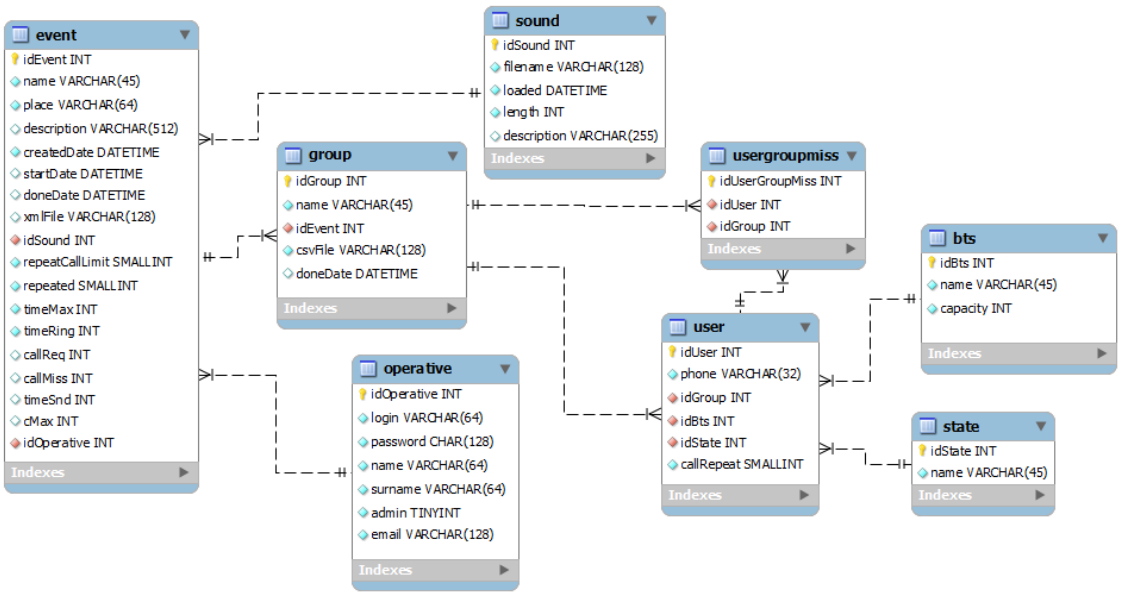
5. view zobrazí aktualizovaná data uživateli

6. view čeká na další požadavky uživatele

4.8 Datová vrstva

Při běhu algoritmu je zapotřebí uchovávat data o stavech jednotlivých požadavků, o rozdělení uživatelů do grup a pro pozdější zobrazení přehledu dokončených akcí také vstupní parametry, které daná úloha využívá. Pro ukládání těchto dat byla zvolena databáze MySQL, protože nabízí vysoký výkon, jednoduchou nasaditelnost a je poskytována pod open-source licencí.

V návrhu datové vrstvy (obrázek 6) jsou zaznamenány cizí klíče, avšak datový engine MyISAM, který je použit z důvodu poskytování co nejvyššího výkonu, nepodporuje integritní omezení pomocí cizích klíčů. Reálné propojení tabulek pomocí vyznačených cizích klíčů musí tedy zajistit sám programátor, například pomocí příkazu join.



Obrázek 6: ER diagram

Popis tabulek:

- Event: tato tabulka uchovává informace o jednotlivých událostech a parametrech potřebných pro jejich běh, obsahuje tyto vazby. Podle atributů doneDate, startDate a createDate se určuje stav, v jakém se daná událost nachází.

- Group: tabulka reprezentující jednotlivé grupy, do nichž jsou ukládáni uživatelé, a jenž jsou používány pro rozesílání hovorů. V atributu csvFile je uložena cesta k .csv souboru, v němž jsou daní uživatele uloženi ve formátu vhodném pro použití v aplikaci SIPp. Atribut doneDate určuje datum provedení dané grupy.
- Sound: informace o jednotlivých .pcap nahrávkách, které jsou přehrávány při navázání hovoru s cílovým telefonním číslem. Pro správné provedení hovoru je nutné, aby byl správně určen atribut length, tedy délka nahrané zprávy.
- Operative: informace o pracovnících krizového centra, kteří mají umožněn přístup do aplikace. Atribut password slouží pro uložení hesla zakódovaného pomocí algoritmu SHA-512, atribut admin určuje, zda má uživatel oprávnění využívat administrátorské funkce.
- User: záznamy cílových uživatelů, kterým jsou směřovány SIP požadavky. Stav jednotlivých uživatelů je určen atributy callRepeat, který určuje počet opakovaných pokusů o provedení hovoru a idState určující, zda uživatel hovor přijal, odmítnul nebo neexistoval.
- State: číselník se stavy, ve kterých se může nacházet uživatel.
- Bts: tabulka, která určuje stanici operátora, ke které je dané telefonní číslo přihlášeno. Tyto informace budou sloužit pro modul ochrany přehlcení BTS.
- UserGroupMiss: vazební tabulka umožňující přiřadit jednoho uživatele více grupám. Je využívána pouze v případě, že uživatel nebyl zastižen a je prováděno nové rozesílání hovoru. Bylo možné využít tuto tabulku i pro rozesílání prvních hovorů, avšak počet záznamů by poté narůstal až příliš rychle a je pravděpodobné, že by docházelo k dlouhému vyhledávání záznamů v této tabulce. Proto byla zvolena možnost, že nejprve je uživatel přiřazen pouze jedné grupě (vazba 1:1) a až pokud je zapotřebí přiřadit jej do nové grupy, je použita tabulka UserGroupMiss.

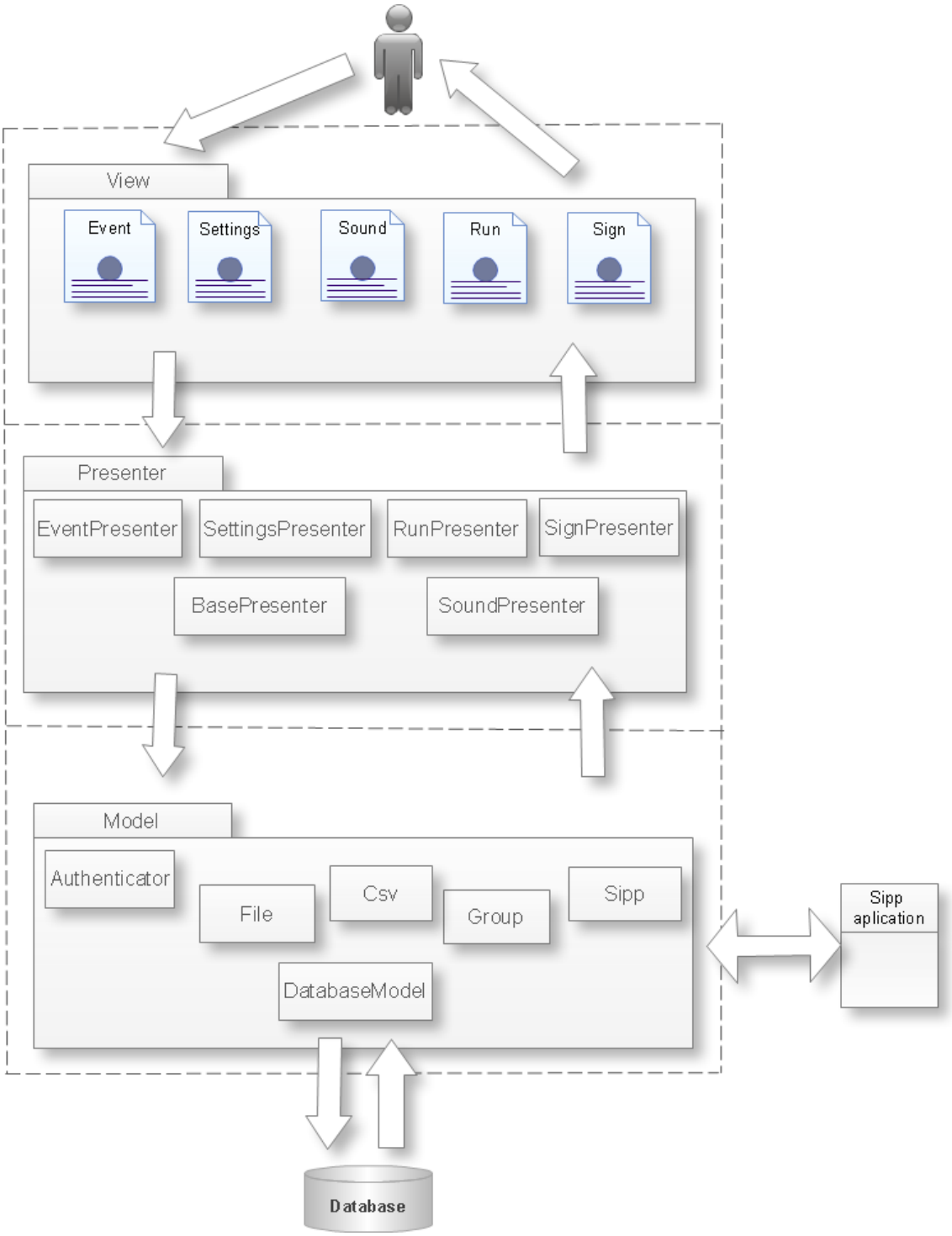
Lineární zápis typů vztahů:

- Event-Sound **N:1**
- Event-Group **1:N**
- Event-Operative **N:1**

- Group-User **1:1**
- Group-UserGroupMiss **1:N**
- UserGroupMiss-User **N:1**
- User-Bts **N:1**
- User-State **N:1**

4.9 Rozvržení tříd

Jednotlivé vrstvy aplikace budou rozvrženy do vrstev modelu MVP. Pohled (View) obsahuje šablony a html soubory generované aplikací ve formátu latte, jenž je defaultní šablonovací systém použitý v Nette frameworku. Tyto šablony zajišťují zobrazování výsledných dat uživateli a umožňují úplné oddělení logiky aplikace. Model obsahuje třídy zajišťující logické operace a výpočty nad daty předávanými presenterům. Zajišťuje také komunikaci s databází a funkce potřebné pro běh aplikace, jako například authentizaci. Presentery jsou třídy nacházející se mezi modelem a pohledem, jenž umožňují přehledně přiřazovat výsledky operací prováděných modelem do umístění jednotlivých stránek. Každá stránka (šablona) má svůj vlastní presenter, a lze tedy přesně určit, které metody modelu jsou na této stránce využity.

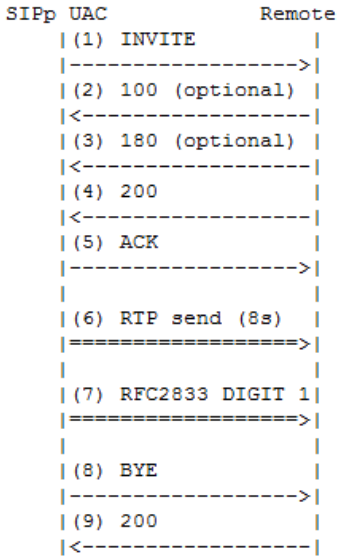


Obrázek 7: Rozvržení tříd

5 Postup implementace

5.1 Tvorba .xml schématu

Aplikace SIPp pro generování SIP požadavků využívá scénáře ve formě .xml schémat, které určují jakým způsobem má probíhat komunikace s uživateli pomocí SIP protokolu. Lze využít buď integrované schémata, která jsou obsažena přímo v aplikaci (zdroj [7]), anebo vytvořit scénář na míru podle potřeby. Základní scénáře, jako například uac_pcap.xml, případně uac.xml mají funkcionalitu velmi podobnou té, jenž je zapotřebí v hlasovém distribučním systému, jelikož jednotlivé nastavení generování hovorů lze upravovat pomocí vkládaných parametrů při spouštění z příkazové řádky. Avšak některé parametry lze změnit pouze v .xml schématu, a proto je nutné vytvořit vlastní scénář. Schéma požadovaného scénáře bude stejné jako schéma integrovaného schématu uac_pcap.xml, které lze vidět na obrázku 8. (zdroj [13], [7], [4])



Obrázek 8: Schéma SIP požadavku

- Na začátku scénáře je nutné nadefinovat .xml hlavičku a název scénáře

```
<?xml version="1.0" ?><scenario name="Generate_calls_pcap">
```

Výpis 1: Začátek scénáře

- Následně započne pokus o navázání spojení zasláním INVITE požadavku a vyčkáním na odpověď. Je důležité nadefinovat kodeky, jenž budou využity pro přehrávání pcap zpráv, v tomto případě PCMU a PCMA.

```
<send retrans="500">
<![CDATA[INVITE sip:[service]@[remote_ip]:[remote_port] SIP/2.0
    Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
    From: sipp <sip:sipp@[local_ip]:[local_port]>;tag=[call_number]
    To: sut <sip:[service]@[remote_ip]:[remote_port]>
    Call-ID: [call_id]
    CSeq: 1 INVITE
    Contact: sip:sipp@[local_ip]:[local_port]
    Max-Forwards: 70
    Content-Type: application/sdp
    Content-Length: [len]
    v=0
    o=user1 53655765 2353687637 IN IP[local_ip_type] [local_ip]
    s=-
    c=IN IP[local_ip_type] [local_ip]
    t=0 0
    m=audio [auto_media_port] RTP/AVP 0 8
    a=rtpmap:0 PCMU/8000
    a=rtpmap:8 PCMA/8000
    a=ptime:20
    a=sendrecv]]>
</send>
```

Výpis 2: Část scénáře zasílající INVITE zprávu

- Pokud má obdržená odpověď kód 100 (Trying) nebo 180 (Ringing) vyčkává scénář, dokud nedorazí odpověď s kódem 200 (OK), a poté zašle potvrzovací ACK zprávu.

```
<send>
<![CDATA[ACK sip:[service]@[remote_ip]:[remote_port] SIP/2.0
    Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
    From: sipp <sip:sipp@[local_ip]:[local_port]>;tag=[call_number]
    To: sut <sip:[field0]@[remote_ip]:[remote_port]>[peer_tag_param]
    Call-ID: [call_id]
    CSeq: 1 ACK
    Contact: sip:sipp@[local_ip]:[local_port]
    Max-Forwards: 70
    Content-Length: 0]]>
```

```
</send>
```

Výpis 3: Část scénáře zasílající ACK zprávu

- Má-li v průběhu scénáře obdržená odpověď kód 503 (Service Unavailable), 404 (Not Found) nebo 486 (Busy Here), pak je zbytek scénáře přeskočen a dojde k ukončení.

```
<recv response="486" optional="true" next="10"></recv>
<recv response="503" optional="true" next="10"></recv>
<recv response="404" optional="true" next="10"></recv>
```

Výpis 4: Část scénáře pro ošetření zpráv pro nedostupnost volaného účastníka

- Po zaslání ACK začne přehrávat nahrávku a nastaví prodlevu tak, aby došlo k přehrání celé nahrávky.

```
<nop>
  <action>
    <exec play_pcap_audio="/home/kolenovsky/g711u.pcap"/>
  </action>
</nop>
<pause milliseconds="9000"/>
```

Výpis 5: Část scénáře přehrávající soubor .pcap

- Po přehrání .pcap souboru je odeslána zpráva BYE a v případě, že je odpověď 200 (OK) je scénář ukončen.

```
<send retrans="500">
<![CDATA[BYE sip:[service]@[remote_ip]:[remote_port] SIP/2.0
      Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
      From: sipp <sip:sipp@[local_ip]:[local_port]>;tag=[call_number]
      To: sut <sip:[field0]@[remote_ip]:[remote_port]>[peer_tag_param]
      Call-ID: [call_id]
      CSeq: 2 BYE
      Contact: sip:sipp@[local_ip]:[local_port]
      Max-Forwards: 70
      Content-Length: 0]]>
</send>
```

Výpis 6: Část scénáře zasílající BYE zprávu

- Na konci scénáře je uveden label na který je přeskočeno, pokud není hovor přijat a nadefinovány časy distribuce zasílání odpovědí.

```
<label id="10"/>
<ResponseTimeRepartition value="10, 20, 30, 40, 50, 100, 150, 200"/>
<CallLengthRepartition value="10, 50, 100, 500, 1000, 5000, 10000"/>
</scenario>
```

Výpis 7: Ukončení scénáře

Pro generování hovorů na určená čísla je nutné použít tzv. injection file, tedy soubor, který umožňuje dynamicky měnit parametry schématu přímo za běhu aplikace. Tento soubor musí být vkládán ve formátu .csv, přičemž řádky reprezentují jednotlivé záznamy a sloupce, oddělené znakem ";" (středník), určují atributy záznamu. První řádek souboru určuje způsob čtení záznamů, tedy zda mají být čteny postupně (SEQUENTIAL), náhodně (RANDOM) nebo podle definice uživatele (USER). Ukázkový soubor example.csv vhodný pro účely hds, by tedy mohl vypadat takto:

```
SEQUENTIAL
776156358
753153682
756116342
```

Výpis 8: Ukázka .csv souboru

Pro potřeby rozesílání je dostačující pouze cílové telefonní číslo, nicméně daly by se do schématu vložit další informace, jako jméno uživatele, případně heslo. Aby bylo možné tento soubor využít, je nutné nadefinovat na jaké místo v .xml schématu mají být hodnoty dosazovány. K hodnotám ze souboru se přistupuje pomocí názvu field a číslem atributu, tedy první atribut má hodnotu field0, druhý field1 atd. Soubor obsahuje hodnoty cílových telefonních čísel, atribut field0 tedy ve schématu nahradí hodnotu service, která defaultně hodnotu cílového čísla uchovává. Je nutné nahradit všechny výskyty proměnné service ve scénáři (tedy v INVITE, ACK i BYE) a upravené zprávy budou tedy vypadat takto:

```
<send>
<![CDATA[INVITE sip:[field0]@[remote_ip]:[remote_port]...

<send>
<![CDATA[ACK sip:[field0]@[remote_ip]:[remote_port]...
```

```
<send>
<![CDATA[BYE sip:[field0]@[remote_ip]:[remote_port]...
```

Výpis 9: Část scénáře nahrazená daty z .csv souboru

Další hodnoty, které nelze předávat pomocí parametrů, jsou název přehrávané .pcap zprávy a její délka. Pro změnu těchto hodnot je nutné, aby aplikace přímo změnila jejich hodnotu v prováděném scénáři. Prakticky je tato operace řešena použitím výchozího scénáře, ve kterém jsou měněné hodnoty nahrazeny unikátními řetězci, které nejsou ve scénáři jinak použity a vygenerováním samostatného scénáře pro každou prováděnou událost. V těchto scénářích jsou vzorové hodnoty (MSG, MSG-LENGTH a RING-TIME) nahrazeny uživatelem nadefinovanými hodnotami.

```
sipp -sf nazev-scenare ip-adresa-PBX
```

Výpis 10: Ukázkový příkaz pro spuštění aplikace SIPp

použité parametry:

- **-m** celkový počet generovaných hovorů
- **-r** počet hovorů za dobu rp (rate period)
- **-rp** doba pro rozesílání hovorů, defaultně 1 sekunda
- **-l** maximální počet simultálních hovorů
- **-aa** automatické potvrzování zpráv typu INFO, UPDATE a NOTIFY
- **-inf** volba injection .csv souboru
- **-trace_msg** ukládání všech přijatých SIP zpráv do logu
- **-trace_err** ukládání chyb do logu

```
sipp -aa -r 500 -t u1 -sf example.xml -inf users.csv 127.0.0.1 -l 10000 -m 500 -i 127.0.0.1 -
    p 5070 -trace_err -trace_msg -trace_counts
```

Výpis 11: Finální příkaz pro spuštění aplikace SIPp spolu s vytvořeným scénářem a .csv souborem

5.2 Tvorba webové aplikace

Tvorba webové aplikace probíhala v Nette Frameworku (verze 2.0), PHP 5.3 pro skriptování na straně serveru a JavaScriptovou knihovnou jQuery ([25])pro skritování na straně klienta. Pro přístup k databázi bylo použito rozšíření PDO (zdroj [15]), které umožňuje využití předpřipravených dotazů (prepared statements), jenž ulehčují práci z hlediska automatické ochrany proti SQL injekci a také zlepšují výkon při opakovaném provádění dotazů. Vzhled a zobrazení dat uživateli je prováděno pomocí html stránek a kaskádových stylů (zdroj [14])

5.2.1 Konfigurace Nette

Zdrojové kódy Nette frameworku byly získány z oficiálních stránek frameworku (zdroj [17]). Aplikace využívá standardní adresářovou strukturu běžnou pro Nette (tzv. sandbox), kdy zdrojové kódy aplikace (adresář application) jsou odděleny od souborů stránek zobrazovaných uživatelům v prohlížeči (adresář www) a přístup k nim je zamezen pomocí souboru .htaccess. V rámci adresáře application jsou rozděleny vrstvy aplikace do adresářů templates (pohledy), presenters (prezentery), models (modely) a config (konfigurační soubory). Před samotným vytvářením zdrojových kódu je nutné nejprve nakonfigurovat nastavení údajů pro přístup k databázi a také tabulky, která bude použita pro ověřování uživatelů při přihlašování do aplikace. Tyto nastavení se provádějí v souboru config.neon (adresář config), přičemž zde také probíhá přidávání služeb (services), které jsou využívány v jednotlivých prezenterech. Další nastavení se týká routování, tedy překladu URL adres do akcí presenteru a naopak. Díky tomuto nastavení není nutné v odkazech na stránkách volat pouze URL adresy, ale také přímo akce presenterů a také zobrazovaná adresa je v URL řádku zobrazována pro uživatele v přívětivém tvaru. Je tedy nutné nastavit tvar URL adresy, na který má být přepsána původní adresa a také název úvodní stránky, která má být zobrazena. Použitá routovací pravidla vypadají takto:

```
$router = $container->router;
$router[] = new Route('index.php', 'Sign:in', Route::ONE_WAY);
$router[] = new Route('<presenter>/<action>[/<id>]', 'Sign:in');
```

Výpis 12: Konfigurace routovacích pravidel

a určují, že soubor, v němž se nachází spouštěcí příkazy pro Nette aplikaci se jmenuje index.php, presenter, který se má zobrazit jako úvodní, se jmenuje Sign (příkaz :in značí akci, jenž se má provést) a tvar URL adresy bude přeložen takto:

původní URL:	hds.cz/index.php?presenter=event&action=view&id=521
nová URL:	hds.cz/event/view/521

Tyto nastavení se provádějí v souboru bootstrap.php a mimoto se zde nastavují cesty k adresářům s dočasnými a www soubory, a také nastavení debugovacího režimu.

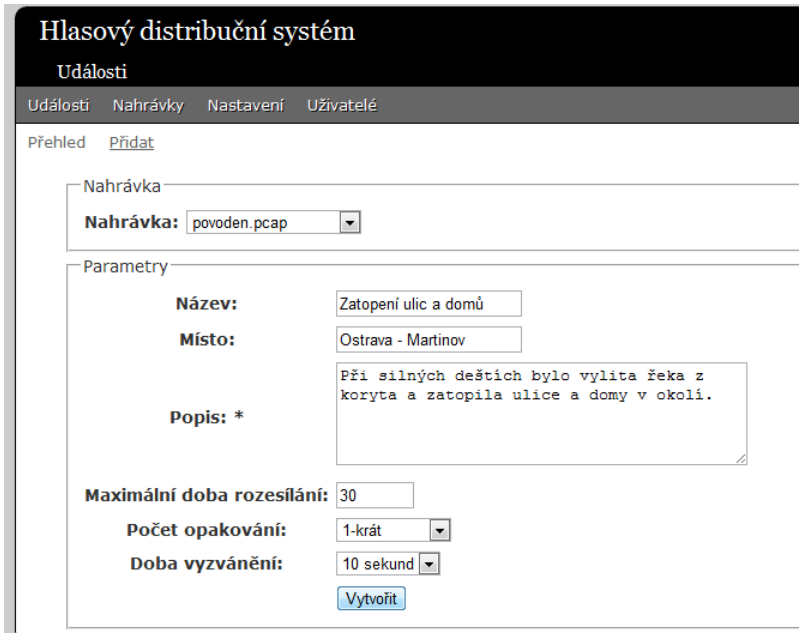
5.2.2 Authentizace

Pro přihlašování uživatelů je využita třída Authenticator, která je již v Nette předpřipravena. Pro její nasazení stačí nadefinovat název tabulky s uživateli a upravit názvy atributů s uživatelským jménem a heslem. V rámci zvýšení bezpečnosti byly v této třídě naimplementovány metody šifrování hesla pomocí algoritmu SHA-512 a tzv. "solení" hesla (zdroj [19]). Metoda solení spočívá v přidání řetězce (tzv. soli) a případné transformaci hesla na jiný tvar před provedením šifrování, což značně znesnadňuje rozšifrování zašifrovaného hesla. Pro ověřování uživatelů při přihlášení je využita tabulka operative a její atributy login a password. Pro zobrazení přihlašovacího formuláře slouží presenter Sign:in.

5.2.3 Příprava události

Pro přípravu a zobrazení událostí je vytvořen presenter Event, který umožňuje vytvářet nové události, přidávat k nim uživatele rozdělené do grup a zobrazovat již provedené události. Po vyplnění formuláře (viz obrázek 9) je vytvořena nová událost a je zobrazeno pole pro nahrání .xml souboru s uživateli. Tito uživatelé jsou rozděleni do grup o velikostech Cmax (hodnotu lze měnit v nastavení aplikace), následně jsou uloženi do databáze, a také do .csv souborů jednotlivých grup. Uživateli jsou poté zobrazeny statistiky počtu grup, uživatelů a předpokládaného času pro rozesílání a v případě potřeby může nahrané uživatele odstranit a provést nové nahrání. Pokud se rozhodne, že je již vše v pořádku, pak může přejít na presenter Run, kde je prováděno samotné rozesílání hovorů. (zdroj [17])]

Formuláře jsou vytvářeny pomocí Nette knihovny UI/Form, která umožňuje definovat validační pravidla a automaticky generuje potřebný kód jak na straně klienta, tak na straně serveru pro případ, že by klient neměl povolen JavaScript. Vkládané hodnoty



Obrázek 9: Přidání události

jsou tedy plně validní a není možné vytvořit událost s neplatnými informacemi. Příklad definice vlastních formulářů, kdy je určeno, že dané pole musí obsahovat číslo a nemůže být nevyplněno:

```
$form = new Form()
$form->addText('timeMax', 'Maximální doba rozesílání:', 5, 12)
    ->addRule(Form::FILLED, 'Je nutné zadat maximální délku.')
    ->addRule(Form::INTEGER, 'Délka rozesílání musí být číslo');
```

Výpis 13: Ukázka vytvoření formuláře s validací (zdroj [17])

5.2.4 Generování hovorů

Nejsložitější částí aplikace je algoritmus provádějící generování hovorů a zpracování výsledků. K jeho provedení slouží presenter Run, který obsahuje metody pro zpracování AJAXových požadavků zasílaných pomocí JavaScriptu (zdroj [12]), což umožňuje rychlou odezvu pro zobrazování informací o průběhu rozesílání hovorů obsluze aplikace. Samotný běh algoritmu je tedy realizován pomocí JavaScriptu, který předává jednotlivé požadavky na provádění hovorů, zpracování výsledků a uložení do databáze na server pomocí asynchronních požadavků. Předávání těchto požadavků probíhá pomocí tzv.

handlů (nebo také subrequestů), kdy název handlu z JavaScriptu je odeslán na server, kde je provedena metoda náležící danému názvu handlu a vrácen výsledek zpět do metody v JavaScriptu, ze které výsledek může předat přímo do HTML stránky nebo předat jiné metodě. Pokud však má být hodnota předávána mezi různými metodami, pak musí být nastaveno synchronní zasílání požadavků, aby průběh kódu vyčkal na návratovou hodnotu z AJAXového požadavku.

Příklad provedení AJAX požadavku kontrolujícího stav procesu:

Klientská část - při volání metody check_process dojde k zaslání požadavku s handlem checkProcess a předání hodnoty proměnné pid. Požadavek je zaslán v asynchronním režimu a po provedení je zpracována proměnná live, která je předána zpět po provedení na straně serveru. Pokud je hodnota live rovna hodnotě 1 (proces stále trvá), pak je znovu zavolána metoda check_process. Pokud live není rovno jedné (proces již neprobíhá), pak dojde k zpracování výsledků metodou group_log a spuštění rozesílání další grupy metodou run_group. K zasílání požadavků dochází s prodlevou 1000ms (zdroj [18], [12]).

```
function check_process(pid, groups, i, missed){
    setTimeout(function(){
                    $.ajax({
                        url: {link checkProcess!},
                        data: 'pid='+pid,
                        async: true,
                        timeout: 5000,
                        success: function(payload) {
                            var status = parseInt(payload.live);
                            if(status == 1){
                                check_process(pid, groups, i, missed);
                            }else{
                                group_log(pid, groups[i], missed);
                                run_group(groups, i+1, missed);
                            }
                        }
                    });
    }, 1000);
}
```

Výpis 14: Ukázka JavaScriptové funkce pro volání ajaxového požadavku

Serverová část - server zpracuje požadavek na handle checkProcess se vstupní proměnnou pid. Ze svého objektu SIPp volá metodu checkProcess a předává ji proměnnou

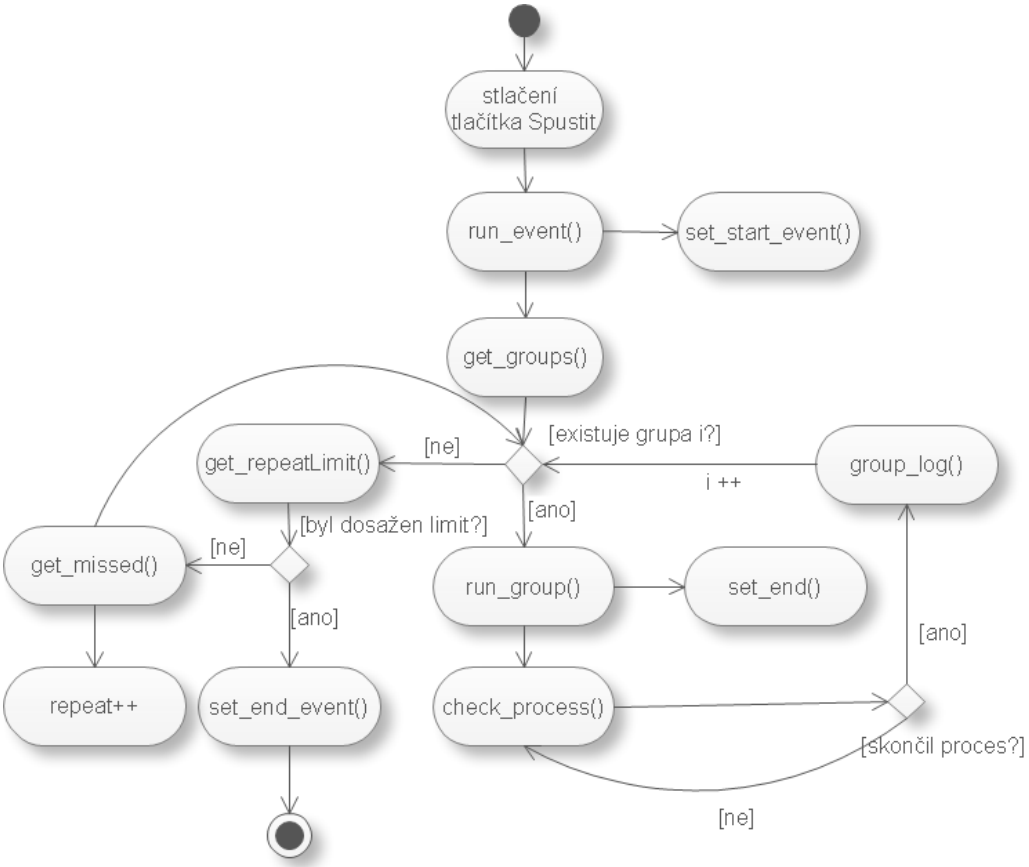
pid, přičemž návratovou hodnotu volané metody přiřazuje do proměnné live, kterou následně odešle zpět do prohlížeče klienta. (zdroj [17])

```
public function handleCheckProcess($pid)
{
        $this->payload->live = $this->sipp->checkProcess($pid);
        $this->sendPayload();
}
```

Výpis 15: Ukázka PHP funkce pro zpracování ajaxového požadavku

Na straně klienta je algoritmus pro rozesílání hovorů zahájen stlačením tlačítka Spustit. Provedením této akce dojde k volání metody run_event(), která zajišťuje nastavení času spuštění události pomocí metody set_start_event() a zobrazení stavového okna s průběhem generování hovorů. Poté pomocí metody get_groups() získá grupy, které náleží dané události a předá je metodě run_group(). Tato metoda provádí zobrazení informací o právě prováděné grupě ve stavovém okně, následně spouští rozesílání aktuální grupy a pomocí metody check_process() kontroluje, zda proces provádějící rozesílání je stále aktivní. Jakmile skončí aktuální proces, je použita metoda group_log(), která provede vyhodnocení hovorů a uloží výsledky do databáze. Zároveň s touto akcí je spuštěna metoda set_end(), která nastaví čas dokončení provádění grupy. Následně jsou prováděny další grupy až do chvíle, dokud nejsou všechny hotovy. Pokud je nastaveno opakované volání nezastiženým uživatelům, je po dokončení všech grup využita metoda get_missed(), která vytvoří nové grupy z nezastižených uživatelů a předá jejich pole do run_group(), kde jsou vytvořené grupy prováděny stejným způsobem jako předešlé, až do dosažení limitu pro opakované volání. Po dokončení všech grup je nastaven čas dokončení události metodou set_end_event() a je uložen log s průběhem události do textového souboru.

Kdykoliv v průběhu provádění události je možné ukončit proces rozesílání pomocí tlačítka Ukončit. Pokud je stlačeno, dojde k volání funkce kill_process(), která ukončí rozesílání aktuální grupy a zabrání započetí grupy nové. Aby bylo zabráněno uživateli přejít na jinou část aplikace, je během rozesílání skryto menu a uživatel tedy nemá možnost přecházet do jiných sekcí aplikace, dokud rozesílání není ukončeno.

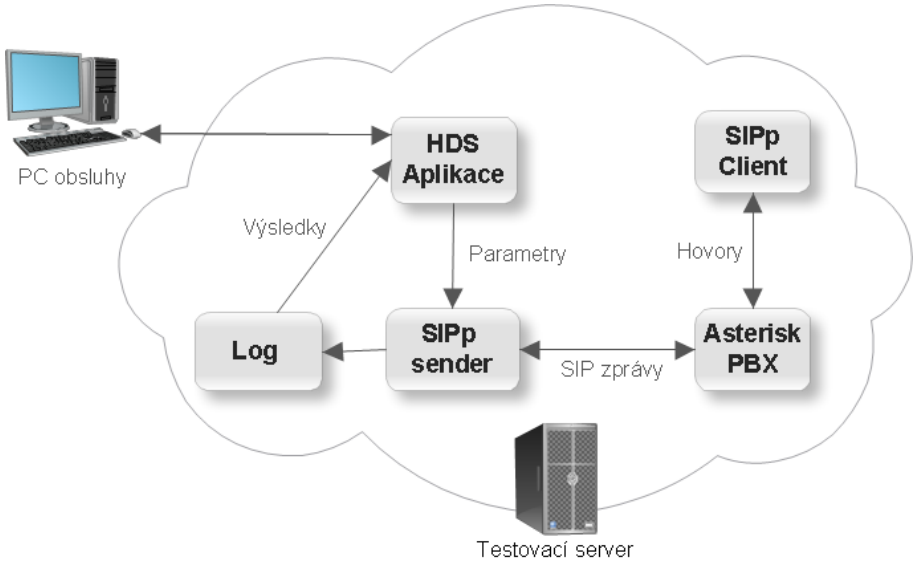


Obrázek 10: Průběh volání jednotlivých metod

6 Testování a porovnání výsledků

Testování funkčnosti distribučního systému probíhalo pomocí dvou softwarových telefonů, které byly registrovány na telefonní účty nakonfigurované v telefonní ústředně asterisk. Funkci telefonů byla realizována pomocí volně dostupných verzí aplikací Ekiga a X-Lite (zdroj [3]). Jako testovací server byl využit Dell poweredge R410, na němž byl nainstalována Linuxová distribuce Debian, jenž byla provozována jako virtuální stroj. Pro základní testování byl použit .xml soubor obsahující předvolby (400,410,420) tří uživatelů, které byly nastaveny pro vytáčení přihlášených uživatelů. Po nahrání seznamu byly vytvořeny 3 grupy a v každé grupě byla přiřazena 1 předvolba, kdy pouze dvě z těchto čísel byly aktivně přihlášeny k ústředně, a jedno bylo neaktivní. Jakmile došlo ke spuštění vytvořené události, bylo postupně spouštěno rozesílání grup, kdy nejdříve začala vyzvánět aplikace X-Lite (s registrovanou předvolbou 400), a jakmile došlo k odmítnutí hovoru a jeho vyhodnocení jako nepřijatého, byl realizován hovor k aplikaci Ekiga (s předvolbou 410). Když byl hovor přijat, začala se přehrávat přednahraná zpráva (g711u.pcap), poté byl hovor po 12 sekundách ukončen a systémem jej vyhodnotil jako přijatý. Třetí předvolba (420) nebyla registrována na žádném aktivním telefonu a systém ji tedy označil jako neexistující. Provádění tohoto testovacího scénáře se třemi grupami trvalo 42 sekund, protože v jedné grupě nebyl hovor přijat a v další dokonce ani nebyla aktivní předvolba uživatele, avšak v reálném provozu s více uživateli by tato doba byla delší, protože někteří účastníci by mohli nechat vyzvánět aplikaci až po maximální dobu 15 sekund a přednahraná zpráva by mohla mít až 30 sekund. V jiném testovacím případě byly všechny tři předvolby zařazeny do jedné grupy a během provádění byly realizovány hovory současně.

Pro porovnání předpokládaných teoretických výsledků maximálního zatížení aplikace bylo zapotřebí simulovat provoz velkého množství uživatelů. Pro tyto účely již nedostačovala simulace pomocí VoIP telefonních aplikací, ale bylo zapotřebí připojit velké množství uživatelů. K tomuto účelu opět posloužila aplikace SIPp, která v tomto případě simuluje přihlášené klienty, avšak aby tito klienti mohli být připojeni, bylo nutné nakonfigurovat je rozdílně oproti účtům provozovaným pomocí softwarových telefonů. Konfigurace klientů je uložena v adresáři aplikace Asterisk v souboru sip.conf a volané předvolby přiřazené daným uživatelům poté v souboru extensions.conf (zdroj [5]). V průběhu prvního testu byli uživatelé nakonfigurováni dynamickým způsobem, kdy atribut host byl nastaven na hodnotu dynamic bez uvedení portu, na kterém má být



Obrázek 11: Schéma testovacího systému

prováděna komunikace. V případě simulací klientů pomocí SIPp byl nastaven atribut host na hodnotu 127.0.0.1 (ip adresa localhostu) a pevně určen port pro komunikaci na hodnotu 5080. Určení portu bylo nutné z důvodu zabránění kolize klientského SIPp s instancí SIPp, jenž provádí rozesílání hovorů a běží na portu 5070. (zdroj [7])

```
[novak]                          [user0]
type=friend                      type=friend
username=novak                   username=user0
host=dynamic                     host=127.0.0.1
secret=147258                    port=5080
insecure=port, invite            insecure=port, invite
context=sipp                     context=sipp
```

Výpis 16: Porovnání konfigurace uživatelů v souboru sip.conf

```
exten => 410,1,Dial(SIP/novak)
exten => 500,1,Dial(SIP/user0)
```

Výpis 17: Konfigurace převoleb v souboru extensions.conf

Do konfiguračních souborů bylo pomocí jednoduchého skriptu generováno a postupně vkládáno 40, 80, 160, 320 a 500 uživatelů (user0-user499), byl vytvořen soubor user-test.xml, který obsahoval předvolby těchto uživatelů, a jenž byl následně nahrán do

aplikace distribučního hlasového systému. Poté byla spuštěna aplikace SIPp a následně také rozesílání hovorů v distribučním systému.

```
sipp -sn uas -i 127.0.0.1 -p 5080
```

Výpis 18: Spuštění aplikace SIPp v klientském módu

Generování hovorů probíhalo v pořádku až do hodnoty 160 požadavků v jeden okamžik. Při vytvoření grupy obsahující 320 předvoleb začalo docházet k nesprávnému generování některých požadavků a správně bylo vygenerováno pouze 166 hovorů. Zbylých 154 bylo vráceno s neočekávanou zprávou 500 (Server internal error), protože byly chybně vygenerovány. Pro ověření byli stejní uživatelé rozděleni do 2 grup s velikostí 160 telefonních předvoleb, a takto generované požadavky byly všechny v pořádku vytvořeny a provedeny. Je tedy zřejmé, že problém byl na straně aplikace SIPp, která při dané konfiguraci nedokázala v jeden okamžik bezchybně generovat více než 166 požadavků.

```
Timestamp: Mon Apr 23 10:30:59 2012

Call-rate(length)   Port    Total-time  Total-calls  Remote-host
150.0(0 ms)/1.000s  5070      17.11 s          320  127.0.0.1:5060(UDP)

Call limit reached (-m 320), 0.000 s period 0 ms scheduler resolution
0 calls (limit 10000)                Peak was 167 calls, after 1 s
0 Running, 320 Paused, 0 Woken up
15 dead call msg (discarded)         0 out-of-call msg (discarded)
1 open sockets

                         Messages  Retrans   Timeout   Unexpected-Msg
  INVITE ----------> 320          0         0
     100 <---------- 167          0         0         150
     180 <---------- 167          0         0         0
     486 <---------- 0            0         0         0
     503 <---------- 3            0         0         0
     404 <---------- 0            0         0         0
     200 <---------- E-RTD1 167 0 0         0         0

     ACK ----------> 167          0
         [ NOP ]
   Pause [    16.0s]             167                   2
     BYE ----------> 165          0         0
     200 <---------- 165          0         0         0
```

Výpis 19: Výpis statistik aplikace SIPp z průběhu generování 320 současných požadavků

Oproti původnímu testování provedenému pracovníky katedry telekomunikací (zdroj ([1, článek konference])) bylo nyní testování prováděno pouze na jednom virtuálním serveru, který prováděl zároveň generování SIP požadavků (SIPp), jejich zpracování (Asterisk) a také simulaci účastníků (SIPp). Dále také nebyly nastaveny stejné parametry aplikace SIPp a je možné, že při odlišném nastavení by aplikace dosahovala lepších výsledků. Pokud by tedy byly jednotlivé aplikace na oddělených serverech a bylo by odladěno spouštění aplikace generující SIP požadavky, narostl by také počet požadavků, které je možné generovat v jeden okamžik a lze předpokládat, že by se blížil hodnotám dosaženým během prvního testování. Pro dosažení lepších výsledků je tedy v budoucnu vhodné doplnit aplikaci o podporu vícevláknového provozu, která by zajistila lepší škálovatelnost systému v závislosti na počtu procesorových jader. Pro budoucí testování je vhodné, že pro úpravu spouštěcích parametrů SIPp v rámci zdrojového kódu aplikace stačí pozměnit jednu proměnnou, což je změna triviální a není kvůli tomu zapotřebí hlubších zásahů.

7 Závěr

Cílem diplomové práce bylo navrhnout a implementovat aplikaci pro distribuci hlasových zpráv, vhodnou jako systém varování obyvatel před katastrofami. Tento cíl byl dosažen, a ačkoliv je určitě možné vytvořený systém v mnoha ohledech vylepšit a rozšířit, tak hlavní funkci rozesílání přednahraných zpráv předdefinovaným účastníkům plní dobře. Pro odladění všech chyb by bylo zapotřebí dlouhodobé testování, které bohužel v rámci vypracování této práce nebylo možné provést, avšak vzhledem k vytvořené dokumentaci a použití přehledné architektury rozšířeného Nette frameworku, by neměl být problém kód aplikace upravit podle potřeby.

Během testování aplikace bylo prověřeno, že vytvořený teoretický model funguje, a je tedy možné systém, který je postaven čistě na open-source technologiích, využít v praxi. Je ovšem nutné optimalizovat parametry aplikace SIPp a provozovat aplikační část na samostatném hardware a přidat modul pro podporu vícevláknového provozu aplikace SIPp. Jakmile bude aplikace otestována a doplněna o potřebné moduly nutné pro využití v praxi, bude možné vyjednávat podmínky s mobilními operátory pro poskytování telefonních čísel v případě výskytu katastrofy.

Během prací na vývoji systému jsem se setkával s různými problémy, které bylo nutné pro správnou funkčnost programu řešit, at'už šlo o nastavení potřebných služeb a oprávnění na platformě linux, tvorba .xml scénáře aplikace SIPp nebo samotnou implementaci webové aplikace v použitém frameworku. Hledání řešení těchto problémů mi pomohlo pochopit fungování pro mě nových technologií a aplikací, které mi budou užitečné v budoucím povolání. Za nejdůležitější získané znalosti beru pochopení funkčnosti telefonní ústředny Asterisk, možnost tvorby scénářů a celkové užití aplikace SIPp a rozsáhlejší seznámení s Nette frameworkem, jenž mi bylo umožněno díky implementaci složitějšího systému. Přínos této práce pro rozvoj mých znalostí a dovedností hodnotím jednoznačně kladně. Její tvorba mi umožnila využít mé znalosti webových technologií, rozšířit je o znalost frameworku, který značně ulehčuje jejich využití. Způsob propojení webové aplikace s jinou, provozovanou lokálně mi hodně rozšířil obzor z hlediska možností využití webových technologií.

8 Reference

[1] Miroslav Vozňák, Jaroslav Zdrálek, Filip Řezáč *Threats to Voice over IP Communications Systems*, Ostrava, 2010.

[2] Miroslav Vozňák *Voice over IP. Vysokoškolská skripta*, VŠB-TUO, 2008.

[3] Stephen P. Olejniczak *VoIP Deployment For Dummies*, For Dummies, 2008

[4] Alan B. Johnston *SIP: Understanding the Session Initiation Protocol*, Artech House, 2009

[5] Leif Madsen, Jim Van Meggelen, Russell Bryant *Asterisk: The Definitive Guide*, O'Reilly Media, 2011

[6] Leif Madsen, Russell Bryant *Asterisk Cookbook*, O'Reilly Media, 2011

[7] Richard Gayraud, Olivier Jaques *SIPp - open-source project [Online]* http://sipp.sourceforge.net/doc/reference.html

[8] Tom Adelstein, Bill Lubanovic *Linux System Administration*, O'Reilly Media, 2007

[9] Mark G. Sobell *Practical Guide to Linux Commands, Editors, and Shell Programming, A (2nd Edition)*, Prentice Hall, 2009

[10] Ben Laurie, Peter Laurie *Apache: The Definitive Guide (3rd Edition)*, O'Reilly Media, 2002

[11] David Powers *PHP Object-Oriented Solutions*, friendsofED, 2008

[12] Bogdan Brinzarea, Cristian Darie *AJAX and PHP: Building Modern Web Applications 2nd Edition*, Packt Publishing, 2010

[13] Erik T. Ray *Learning XML, Second Edition*, O'Reilly Media, 2003

[14] Jon Ducket *HTML and CSS: Design and Build Websites*, Wiley, 2011

[15] Dennis Popel *Learning PHP Data Objects: A Beginner's Guide to PHP Data Objects, Database Connection Abstraction Library for PHP 5*, Packt Publishing, 2007

[16] Matt Doyle *Beginning PHP 5.3*, Wrox, 2009

[17] Nette Foundation, David Grundl *Nette framework [Online]* http://doc.nette.org

[18] David Sawyer McFarland *JavaScript & jQuery: The Missing Manual*, Pogue Press, 2011

[19] Bryan Sullivan, Vincent Liu *Web Application Security, A Beginner's Guide*, McGraw-Hill Osborne Media, 2011

[20] The PHP Group *Php [Online]* http://www.php.net/docs.php

[21] David Grundl *Model View Presenter [Online]* http://doc.nette.org/cs/presenters

[22] Pawan Vora *Web Application Design Patterns (Interactive Technologies)*, Morgan Kaufmann, 2009

[23] Paul DuBois *MySQL (4th Edition)*, Addison-Wesley Professional, 2008

[24] *MVP [Online]* http://www.iba.muni.cz/intranet/res/image/webstudio/mvp.png

[25] *jquery.com [Online]* The jQuery Foundation